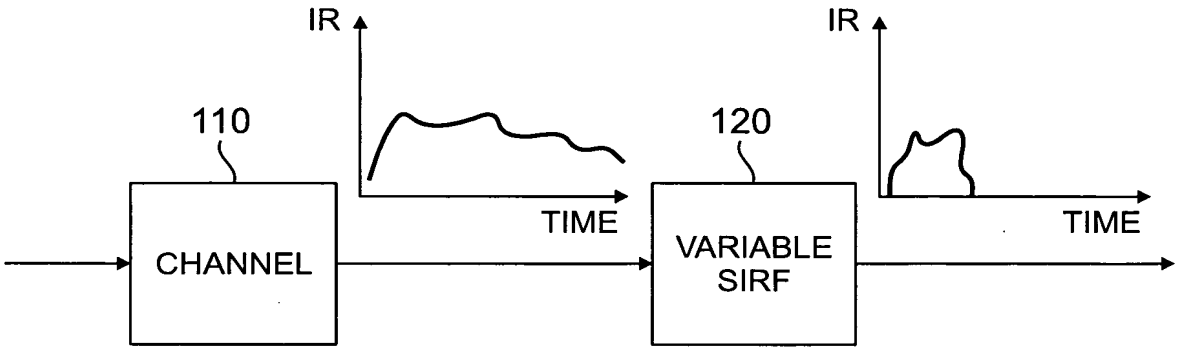

IR
110
TIME
CHANNEL
IR
120
TIME
VARIABLE SIRF

FIG. 1

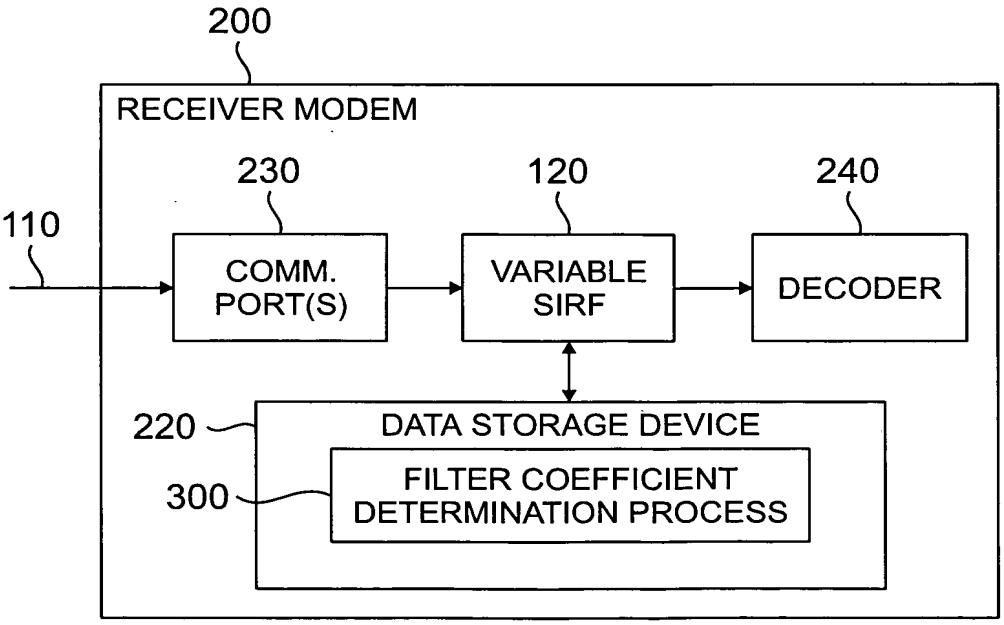

200
RECEIVER MODEM
110
230
COMM. PORT(S)
120
VARIABLE SIRF
240
DECODER
220
DATA STORAGE DEVICE
300
FILTER COEFFICIENT DETERMINATION PROCESS

FIG. 2

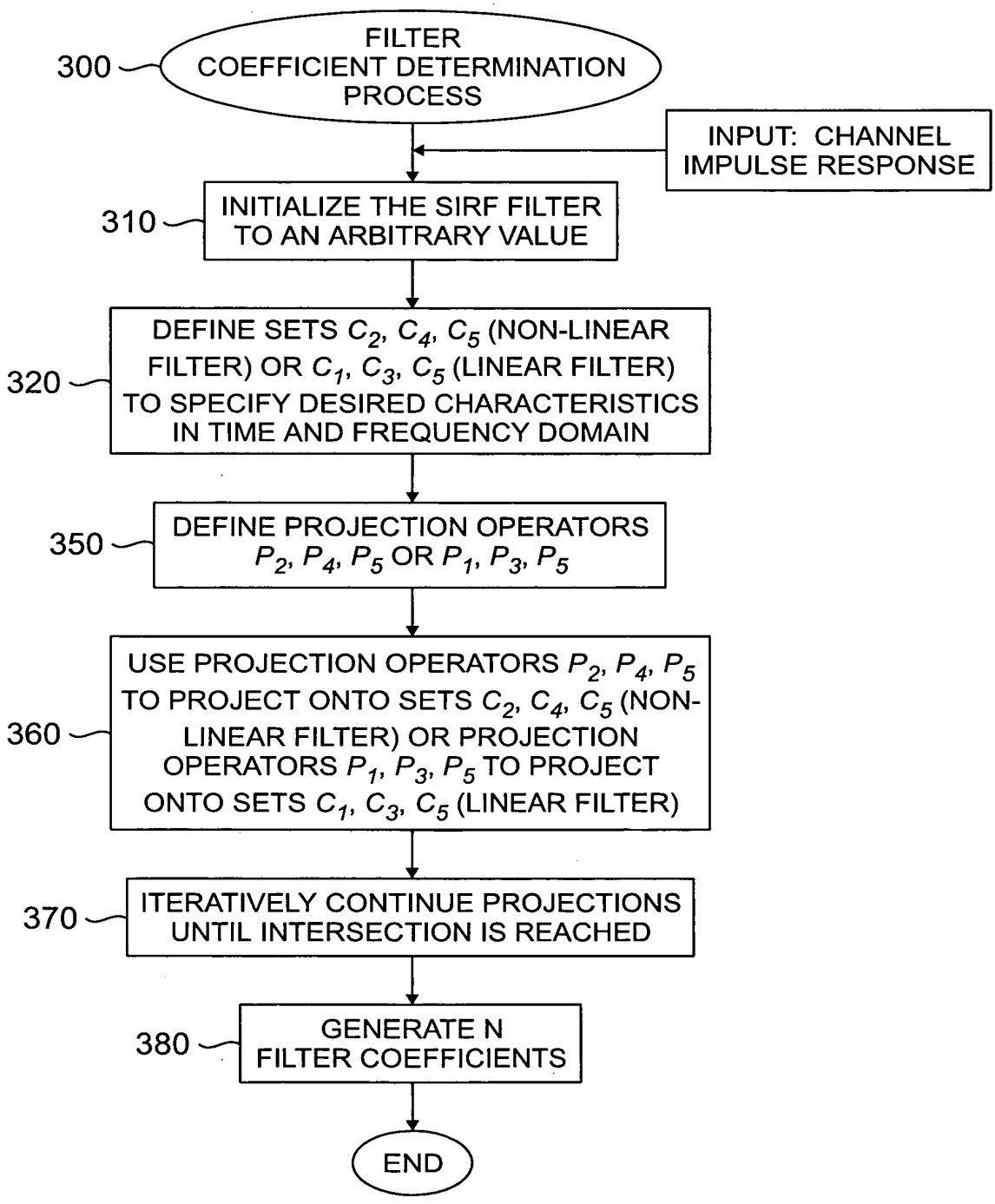

300
FILTER COEFFICIENT DETERMINATION PROCESS
INPUT: CHANNEL IMPULSE RESPONSE
310
INITIALIZE THE SIRF FILTER TO AN ARBITRARY VALUE
320
DEFINE SETS C_2, C_4, C_5 (NON-LINEAR FILTER) OR C_1, C_3, C_5 (LINEAR FILTER) TO SPECIFY DESIRED CHARACTERISTICS IN TIME AND FREQUENCY DOMAIN
350
DEFINE PROJECTION OPERATORS P_2, P_4, P_5 OR P_1, P_3, P_5
360
USE PROJECTION OPERATORS P_2, P_4, P_5 TO PROJECT ONTO SETS C_2, C_4, C_5 (NON-LINEAR FILTER) OR PROJECTION OPERATORS P_1, P_3, P_5 TO PROJECT ONTO SETS C_1, C_3, C_5 (LINEAR FILTER)
370
ITERATIVELY CONTINUE PROJECTIONS UNTIL INTERSECTION IS REACHED
380
GENERATE N FILTER COEFFICIENTS
END

FIG. 3

400

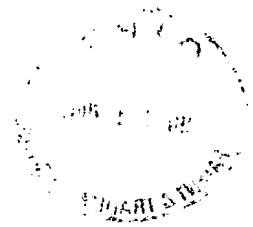
410

```
length = 64; //fast fourier transform length

delta = .1; %input ('Enter stop band tolerance:  ');
beta = 0.2; %input ('Enter pass band tolerance:  ');
wp = 8;
ws = 7;
mu = 28;  %CP length
load channel_impulse_resp.dat;   %load channel impulse response
N = 19;     % sirf filter length
ch = channel_impulse_resp;
mm = max(abs(ch));
ch = 2047*ch./mm;
NN = size(channel_impulse_resp);
M = NN(1);
for i = 1:mu
    mv(i) = 1;   %initialize mv
end
C = conv(mv,abs(ch));
[Y,I] = max(abs(C));   %find max which is beginning of GI (CP)
```

430

```
% construct the channel impulse response matrix
for i = 1:M
    for j = 1:N
        if i - j + 1 <= 0
            break;
        else
            H(i,j) = ch(i - j + 1);
        end
    end
end
n = 1;
for i = M + 1:M + N - 1
    n = n + 1;
    k = 0;
    for j = n:N
            k = k + 1;
            H (i,j) = ch(M - k + 1);
    end
end

for i = 1:mu
    mv(i) = 1;
end
C = conv(mv,abs(ch));
[Y,I] = max(abs(C));
```

FIG. 4A



440

```
for i = 1:mu
    mv(i) = 1;
end
C = conv(mv,abs(ch));
[Y,I] = max(abs(C));
```

445

```
% initialize the SIRF filter to an arbitrary value
c = [  -0.0593
        0.01047
       -0.062386
        0.02418
       -0.065114
        0.030031
       -0.039083
        0.13789
       -0.10266
       -0.014681
       -0.11224
        0.0546
       -0.12642
        0.18608
       -0.020895
        0.38407
       -0.30117
       -0.37885
        0.42326];
```

FIG. 4B

450

```
g = fft(c,length);
f = abs(g);
lamda = 1;
tol = 50;
cold = sum(abs(c));
```

460

```
%begin iteration
for ii = 1:20   % 20 is the number of iterations
%   projection on the set C2
    for m = wp:length/2
        if ( f(m) > (1 + beta))
            gg = g(m);
            g(m) = (1 + beta) / f(m)*complex(real(g(m)),imag(g(m)));
            g(m) = gg + lamda*(g(m) - gg);
        end
        if ( f(m) < (1 - beta))
            gg = g(m);
            g(m) = (1 - beta) / f(m)*complex(real(g(m)),imag(g(m)));
            g(m) = gg+lamda*(g(m) - gg);
        end
    end

for m = 1:ws
        if (f(m) > delta)
            gg = g(m);
            g(m) = delta / f(m)*complex(real(g(m)),imag(g(m)));
            g(m) = gg+lamda*(g(m) - gg);
        end
    end

for m = 2:length/2
    g(m + length/2) = conj(g(length/2 - m + 2));
end
cr = real(ifft((g),length));   %transform to time domain
```

FIG. 4C

```
470  % time domain projection on C4 set
     for i = 1:N
         c(i) = cr(i);  %project into the set C4
     end

     % time domain projection on C5 set
     for n = l - mu - 10:l - mu
         norm = 0;
         prod = 0;
         %prod1 = 0;
         for i = 1:N
             norm = norm + H(n,i) ^ 2;
             prod = prod + c(i)*H(n,i);
             %prod1 = prod1 + c(i)*ch(n - i + 1);
         end
         if (prod > tol)
             for nn = 1:N
                 cc = c(nn);
                 c(nn) = c(nn) + ((tol - prod /norm) * H(n,nn);
                 c(nn) = cc + lamda*(c(nn) - cc);
             end
         end
         if (prod < -tol)
             for nn = 1:N
                 cc = c(nn);
                 c(nn) = c(nn) + ((-tol - prod /norm) * H(n,nn);
                 c(nn) = cc + lamda*(c(nn) - cc);
             end
         end
     end

475-1 for n = l;l + 100
         norm = 0;
         prod = 0;
         for i = 1:N
             norm = norm + H(n,i) ^ 2;
             prod = prod + c(i)*H(n,i);
         end
         if (prod > tol)
             for nn = 1:N
             cc = c(nn);
             c(nn) = c(nn) + ((tol - prod) /norm) * H(n,nn);
             c(nn) = cc + lamda*(c(nn) - cc);

             end
         end
         if (prod < -tol)
```

FIG. 4D

```
475-2                for nn = 1:N
                         cc = c(nn);
                         c(nn) = c(nn) + ((-tol - prod) /norm * H(n,nn);
                         c(nn) = cc + lamda*(c(nn) - cc);
                     end
                 end
             end
             ss = abs(cold - sum(abs(c)))
             cold = sum(abs(c));
             g = fft(c,length);   %transform to frequency domain
             f = abs(g);
         end

480      for i = 1:N
             sirf(i) = c(i);   %sirf will hold the SIRF filter coefficients
         end
```

FIG. 4E

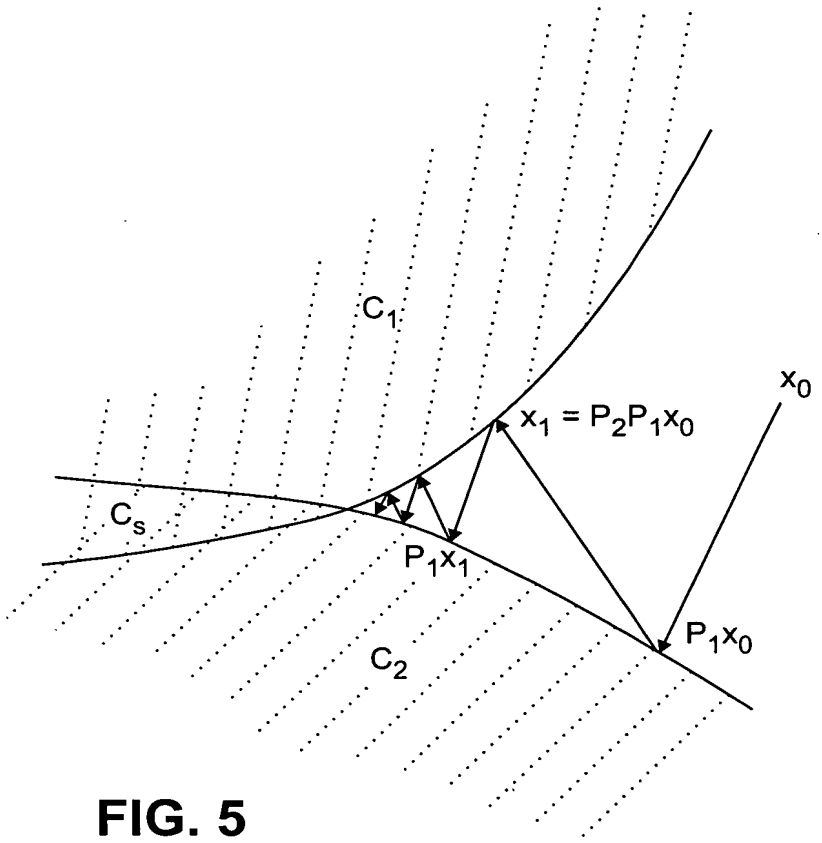


FIG. 5